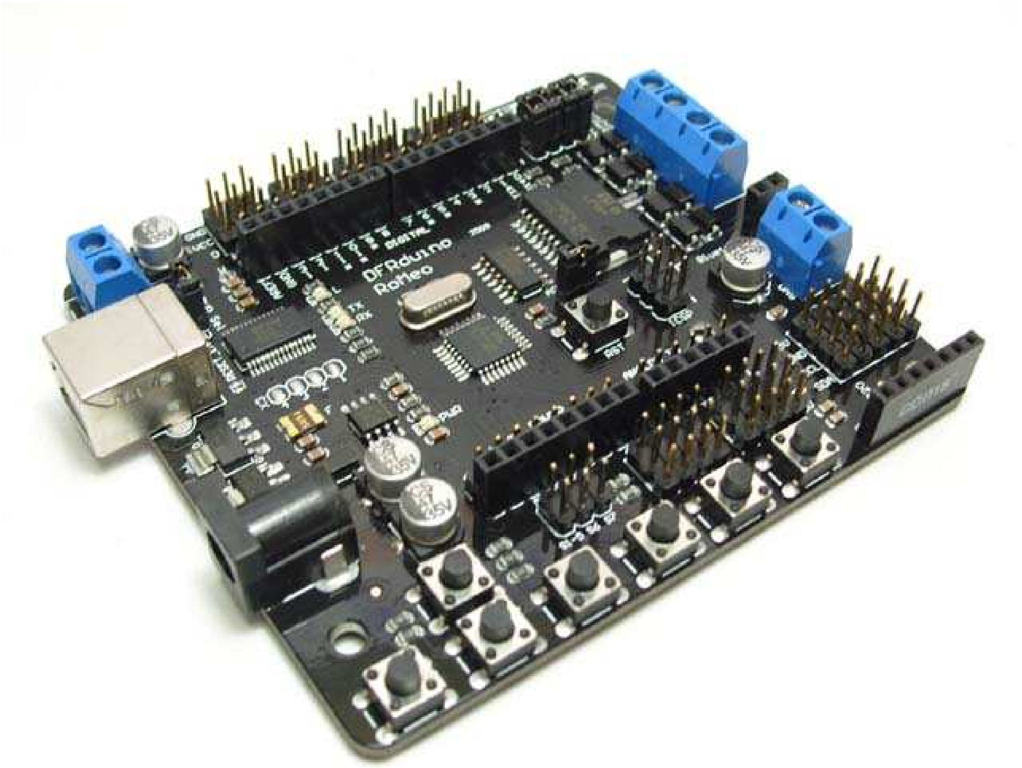
DFRduino RoMeo Users Manual



Dreamfactory 梦工厂

DFRduino RoMeo Users Manual

TEL:（北京总部） 庄先生 010-60899387
（成都办事处）卫先生 15902808530
（上海办事处）桑先生 13774201234

DFRduino RoMeo

A. 注意！在没有认真阅读本说明之前，请勿给模块加电！错误接线将导致模块永久性损坏或烧毁微控制器。

B. 注意！请认真查看引脚功能说明，正确接线！请勿将电源反接，否则将导致模块永久性损坏。

C. 注意！请勿使用超出额定电压的电源！保证电源的稳定，如果出现高压脉冲，可能会导致微控制器永久性损坏。

D. 注意！本产品无防水防潮功能，请在干燥环境下保存或使用！不可将重物堆积在上面。

概 述

什么是 RoMeo?

RoMeo 是一块基与 Arduino 开放原始代码的 Simple i/o 平台，並且具有使用类似 java,C 语言的开发环境。让您可以快速使用 Arduino 语言与 Flash 或 Processing…等软件，作出互动作品。RoMeo 可以使用开发完成的电子元件例如 Switch 或 Sensors 或其他控制器、ＬＥＤ、步进电机或其他输出裝置。RoMeo 也可以独立运作成为一个可以跟软件沟通的平台，例如说：flash processing Max/MSP VVVV 或其他互动软件…

Arduino 开发ＩＤＥ界面基于开放原始码原则，可以让您免费下载使用开发出更多令人惊奇的互动作品或机器人作品。

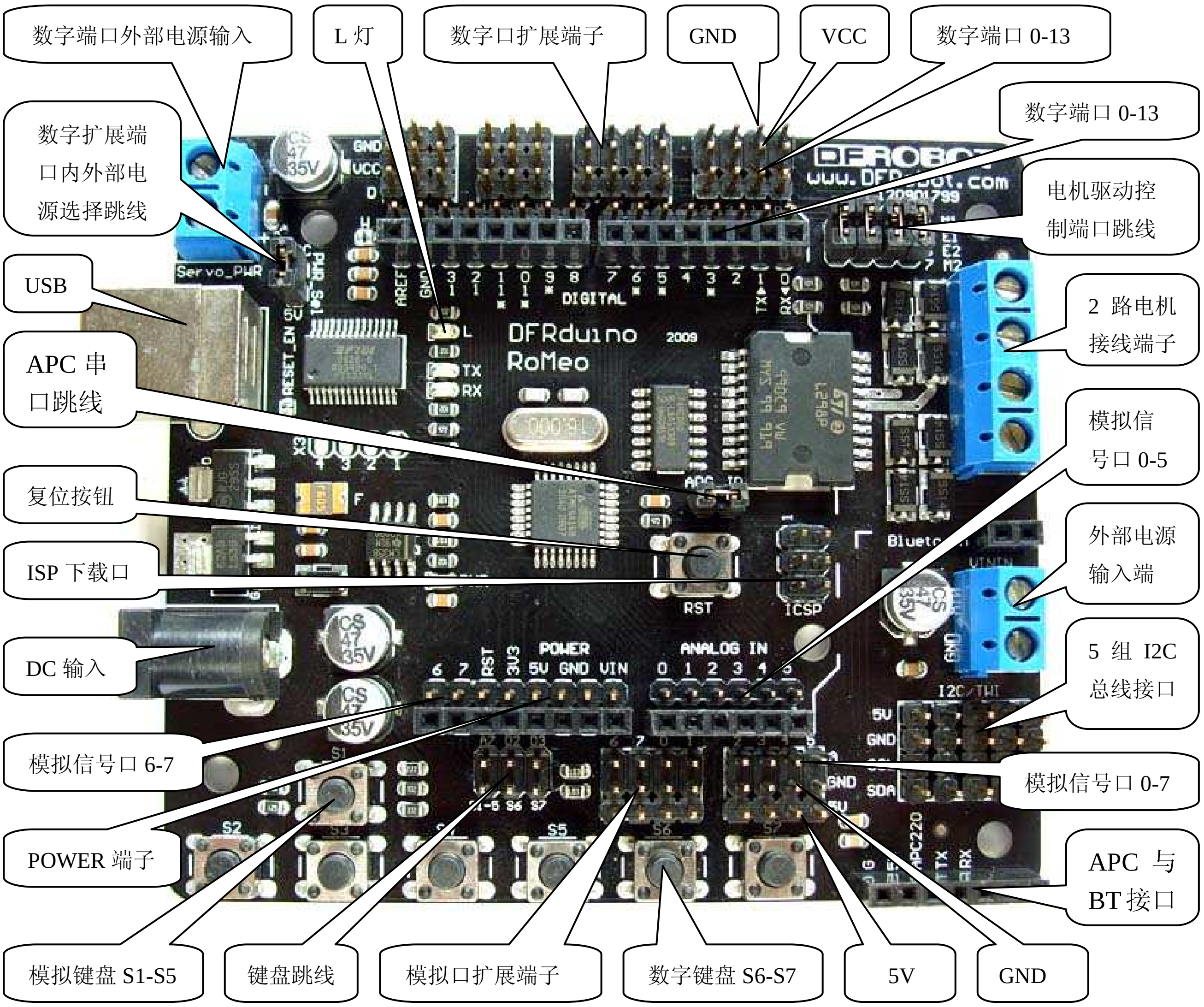
RoMeo 不但有完整的 Arduino 功能，还集成了 2 路电机驱动、无线数传、数字与模拟 IO 扩展口、I2C 总线及键盘输入等接口，并和 Arduino Nano 相同，比传统的 Arduino 多 2 路模拟输入端口。

性能描述

1. Digital I/O 数字输入/输出端 0~13。
2. Analog I/O 模拟输入 0~7。
3. 支持 USB 接口协议。
4. 支持 USB 供电与外部供电自动切换。
5. 支持 ISP 下载功能。
6. 支持单片机 TX/RX 端子。
7. 支持 AREF 端子。
8. 支持数字端、模拟端插针和插孔端子。
9. 集成 APC220 无线数传和 DF-Bluetooth 蓝牙模块接口，并且有 TX 选择跳线选通。
10. 支持 5 组 I2C 总线接口。
11. 支持 2 路电机驱动，峰值电流 2A，4 个控制口使用跳线切换。
12. 支持 7 个按键输入，5 个 A/D 模拟按键(模拟口 7)，2 个数字按键(Pin2，Pin3)，均使用跳线切换。
13. 数字口扩展接口支持单独外部供电或内部 5V 供电，使用跳线切换。
14. 支持六組 PWM 端子(Pin11,Pin10,Pin9、Pin6,Pin5,Pin3)。
15. 输入电压：接上 USB 时无须外部供电或外部供 7V~12V DC 输入。
16. 输出电压：5V DC 输出和 3.3V DC 输出 和 外部电源直接输出。
17. 采用 Atmel Atmega168-20AU 单片机（RoMeo 168 版）或 Atmega328-20AU 单片机（RoMeo 328 版）。
18. RoMeo 大小尺寸：宽 90mm X 高 80mm。

DFRduino RoMeo 的使用

1． 认识 DFRduino RoMeo



2． 集成开发环境的使用：

RoMeo 可直接使用 Arduino 0012 以上版本的集成开发环境，以下简称为 IDE，可到这里下载 http://arduino.cc/en/Main/Software，IDE 中的硬件请选择 Arduino Nano。

Arduino 语言的语法请参考官方网站：http://arduino.cc/en/Reference/HomePage 。

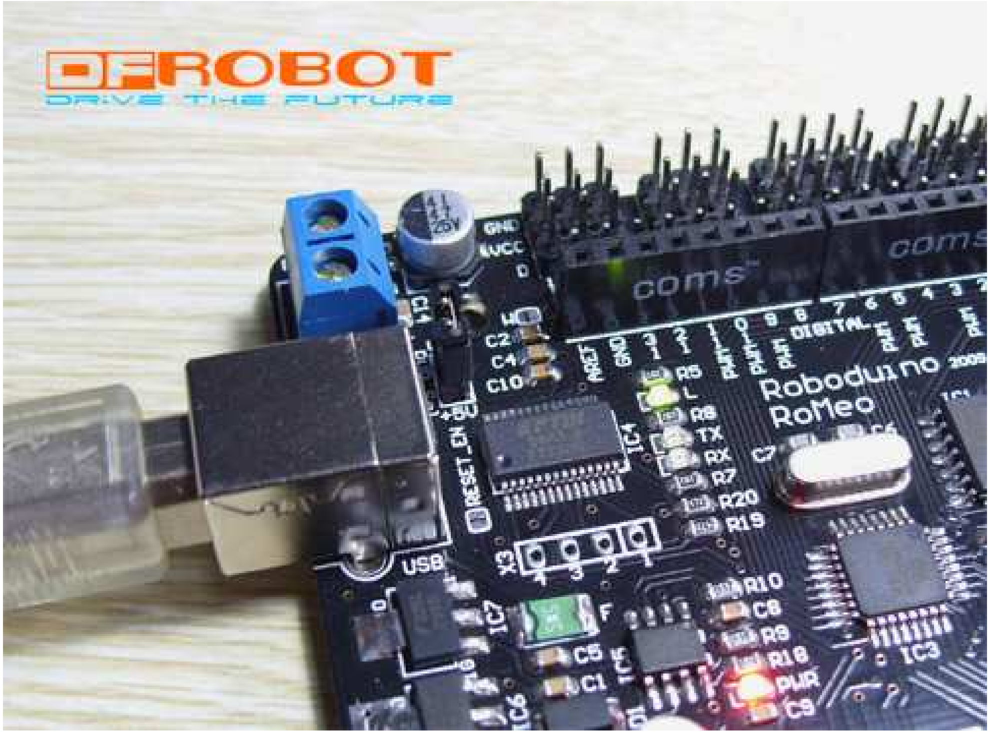
IDE 的使用请参考：http://www.roboticfan.com/blog/user_2005/1229/archives/2008/20084292032.shtml

3． 数字端口的使用：

A． L 灯实验

RoMeo 和 Arduino 一样，有一个使用 Pin13 数字端口控制的板载 LED（L），那么首先做一个数字端口控制一个 LED 闪烁的实验。

将 USB 电缆与 RoMeo 连接后，打开 0012 以上版本的 IDE，硬件选择 Arduino Nano。



演示代码：

```
int ledPin = 13;                    //  定义 LED 为  pin 13
void setup()
{
  pinMode(ledPin, OUTPUT);          //  设置数字端口 13 为输出模式
}
void loop()
{
  digitalWrite(ledPin, HIGH);     //  数字端口置高，LED 亮
  delay(1000);                      //  延时
  digitalWrite(ledPin, LOW);      //  数字端口置低，LED 灭
  delay(1000);                      //  延时
}
```

B．按键（Button）实验

RoMeo 集成了 7 个按键 S1～S7，其中有 2 个按键（S6、S7）是使用数字端口控制。

数字按键 S6 和 S7 使用跳线选通，见下图红圈内，当标有 S6 或 S7 的插针插上短路帽后即表示该按键选通，拔掉则表示断开控制。按键 S6 使用数字端口 Pin2，S7 使用数字端口 Pin3 控制，选通 S6 和 S7 后数字口 Pin2 和 Pin3 将被占用。

下面做一个单按键实验，使用按键 S6，按下放开后 L 灯亮，再按一次 L 灯灭。

演示代码：

```
int ledPin = 13;                       //  定义 LED 为  pin 13
int key_s6 = 2;                         //  定义 S6  为  Pin2
int val=0;
void setup()
{
   pinMode(ledPin, OUTPUT);          //  设置数字端口 13 为输出模式
   pinMode(key_s6, INPUT);           //  设置数字端口 2 为输入模式
}
void loop()
{
    if(digitalRead(key_s6)==0)       //查询 S6 有没有按下
       {
           while(!digitalRead(key_s6));
           val++;

       }
    if(val==1)
```

```
        {
            digitalWrite(ledPin, HIGH);     //  数字端口置高，LED 亮
        }
    if(val==2)
        {
            val=0;
            digitalWrite(ledPin, LOW);      //  数字端口置低，LED 灭
        }
}
```

再做一个双按键实验，按下按键 S6　L 灯亮，按下按键 S7　L 灯灭。

演示代码：

```
int ledPin = 13;                    //  定义 LED 为  pin 13
int key_s6 = 2;                      //  定义 S6  为  Pin2
int key_s7 = 3;                      //  定义 S7 为  Pin3
void setup()
{
   pinMode(ledPin, OUTPUT);           //  设置数字端口 13 为输出模式
   pinMode(key_s6, INPUT);            //  设置数字端口 2 为输入模式
   pinMode(key_s7, INPUT);            //  设置数字端口 3 为输入模式
}
void loop()
{
     if(digitalRead(key_s6)==0)       //查询 S6 有没有按下
        {
             digitalWrite(ledPin, HIGH);     //  数字端口置高，LED 亮
        }
     if(digitalRead(key_s7)==0)     //查询 S7 有没有按下
        {
             digitalWrite(ledPin, LOW);       //  数字端口置低，LED 灭
        }
}
```

C．舵机实验

RoMeo 数字端口扩展端子是按照舵机的线序设计的，所以可以直接控制舵机。接一个舵机时由于电流不大，所以可以暂时使用 RoMeo 上的 5V 供电，但**切记不能使用 USB 供电**，这时需要插上外部直流电源（DC7～12V），如下图所示，RoMeo 上同时插了 USB 电缆和外部直流电源，有人可能会问“不怕烧 USB 吗？”，答案是不怕，因为当外部输入电源电压大于 6.5V 时，RoMeo 会自动从 USB 供电切换为外部电源供电，同时 USB 与外部电源完全隔绝，所以不会烧 USB 接口。

Arduino 0012-0016 版本的 IDE 内会自带一个舵机库文件，不过该库文件只能使 Pin9 和 Pin10 控制舵机，如下图舵机连接到 Pin9 上：

演示代码：

```
#include <Servo.h>
Servo myservo;     //  定义舵机对象，
int pos = 0;
void setup()
{
   myservo.attach(9);   //舵机接到 Pin9 上
}
void loop()
{
   for(pos = 0; pos < 180; pos += 1)          //从 0 度转到 180 度，步进 1 度
   {
      myservo.write(pos);                      //改变舵机度数
      delay(15);                               //  延时
   }
   for(pos = 180; pos>=1; pos-=1)             //从 180 度转到 1 度，步进 1 度
   {
      myservo.write(pos);                      //改变舵机度数
      delay(15);                               //  延时
   }
}
```

使用 Arduino 0017-0018 可以让 RoMeo 和其他 Arduino 控制器的 Pin2-13 端口控制 12 个舵机。当 RoMeo 接多个舵机时，就不能使用 RoMeo 上的 5V 供电了，这时就需要从数字端口外部电源输入端供电，电源电压为 5V。如下图

所示使用的 5V 开关电源供电，也可以用 4-5 节充电电池供电。

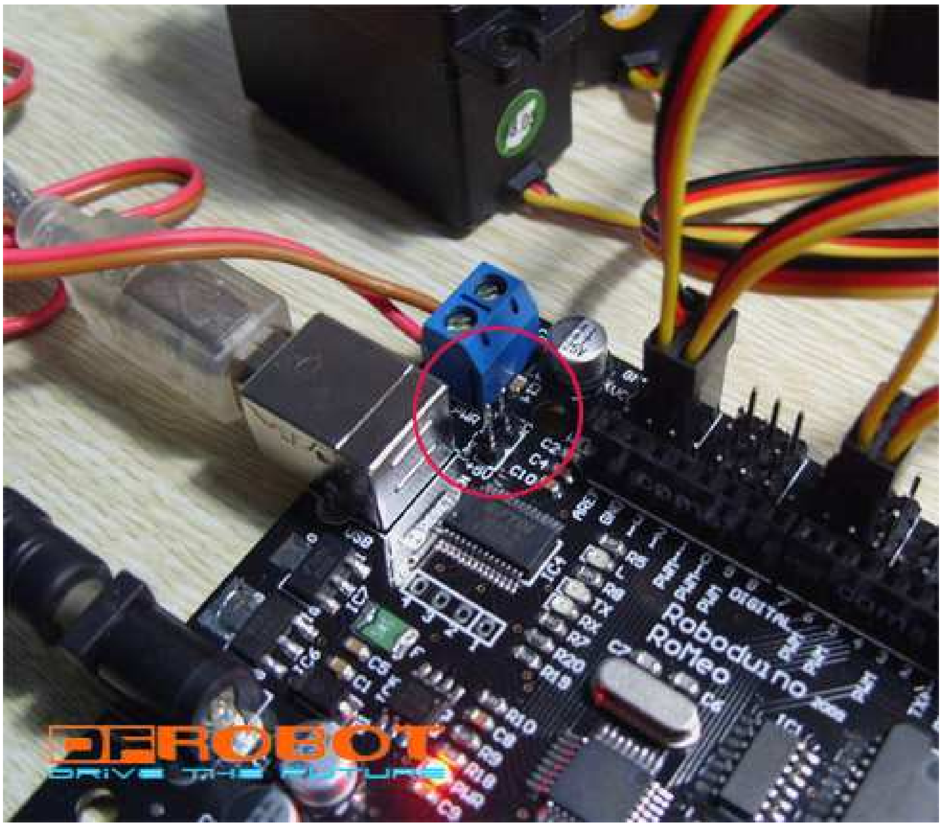
演示代码：

```
#include <Servo.h>
Servo servo1;
Servo servo2;
Servo servo3;
void setup()
{
  servo1.attach(5);              //定义舵机 1 控制口为 Pin5
  servo2.attach(11);             //定义舵机 2 控制口为 Pin11
  servo3.attach(12);             //定义舵机 3 控制口为 Pin12
  Serial.begin(19200);//设置波特率
  Serial.print("Ready");
}
void loop()
{
  static int v = 0;
  if ( Serial.available()) {
    char ch = Serial.read();          //读取串口数据
    switch(ch) {
      case '0'...'9':
        v = v * 10 + ch - '0';   //字符换算成 10 进制
        break;
      case 'a':                  //如果数据后带 a，则表示是 servo1 的数据，比如串口发送 85a
        servo1.write(v);
        v = 0;
        break;
```

```
        case 'b':           //如果数据后带 b，则表示是 servo2 的数据，比如串口发送 90b
          servo2.write(v);
          v = 0;
          break;
        case 'c':           //如果数据后带 c ，则表示是 servo3 的数据，比如串口发送 180c
          servo3.write(v);
          v = 0;
          break;
      }
   }
}
```

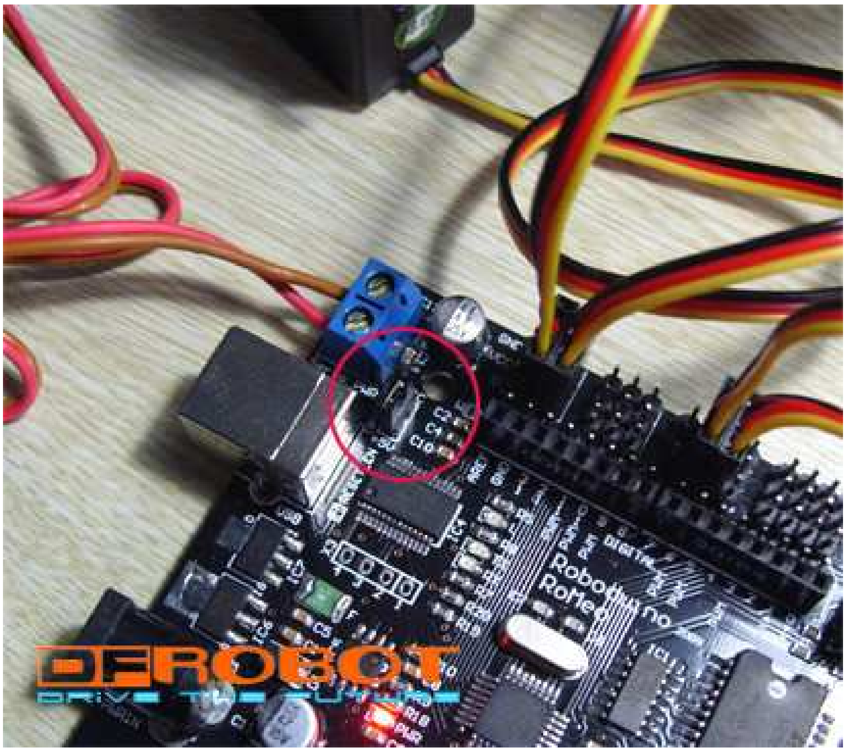
程序功能：通过 IDE 中的串口助手，发送舵机度数控制舵机转动的角度，比如发送字符 85a 表示舵机 1 转动 85 度，发送字符 180c 表示舵机 3 转动 180 度。

在使用舵机过程中会遇到电源供电的问题，RoMeo 上的供电接口与跳线又显的有些复杂，怎么合理的设置才能避免烧掉 RoMeo 和 USB 接口呢，下面我们就详细的介绍一下供电的方式与设置。



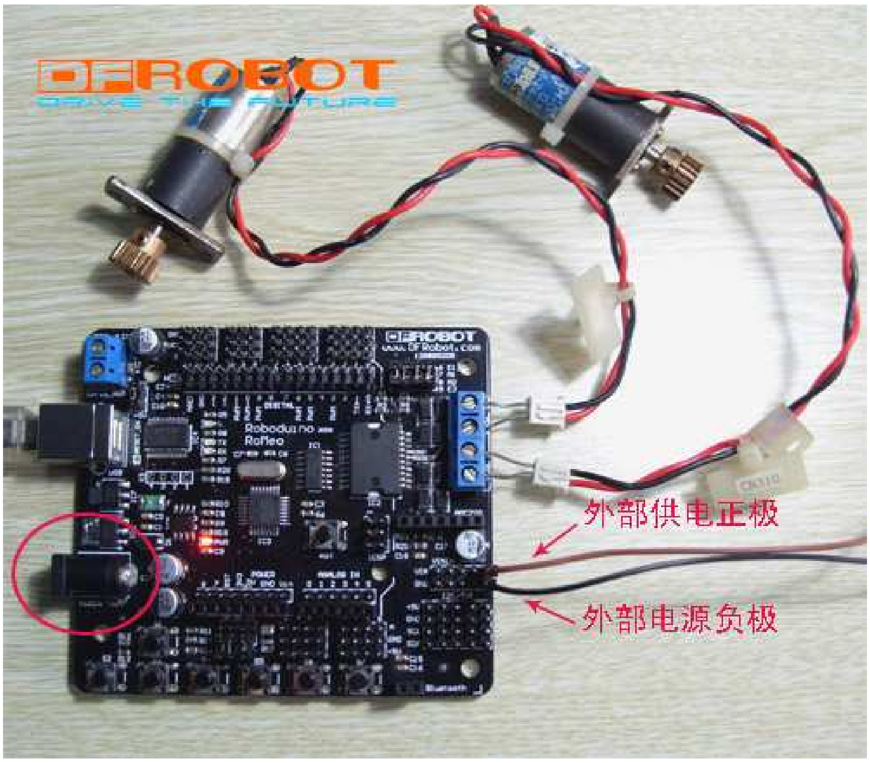
如上图所示，当使用数字外部供电端给舵机供电时，同时 RoMeo 也在用 USB 电缆和外部直流电源供电，那么这时红圈内的内外部电源切换跳线短路帽就要拔掉，逻辑电源和舵机电压分开，防止大电流负载对处理器的干扰，工作更稳定。

当 RoMeo 也要想使用数字端口外部供电时，首先必须保证该电源的电压为 5V，同时需要拔掉 USB 电缆和外部直流电源供电，如上图红圈中所示。然后插上内外部电源切换跳线短路帽，如下图红圈中所示。这时整个 RoMeo 都使用数字端口外部供电了。不推荐使用这种供电方式。

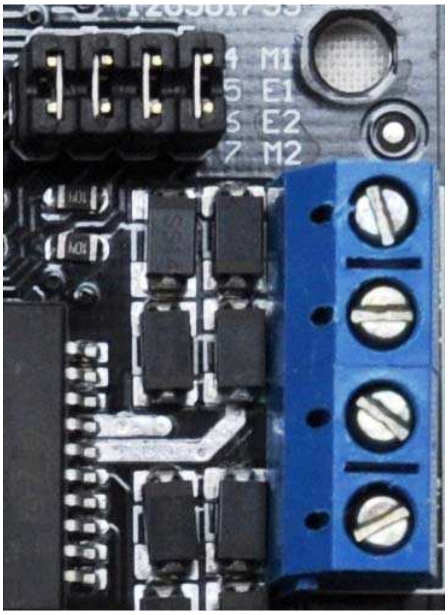


D.　电机驱动实验

RoMeo 上集成了 2 路电机驱动，这是为了让机器人爱好者节约大量制作硬件的时间，而把开发重点放在软件上。电机驱动电路采用 L298 芯片，峰值电流可达 2A。当使用电机驱动时，又会涉及到电源供电问题，如下图所示的接线方法，红圈内的外部直流电源供电就要改到右边的插针上，VIN 接正极，GND 接负极。



电机驱动电路控制端使用短路跳线选通，用的时候接通，不用就断开。



从上图中可以看到，电机控制端口使用数字端口 Pin4、5、6、7，Pin4 是 M1 电机的正反转控制端，Pin5 是 M1 电机的使能端，可以进行 PWM 调速；Pin6 是 M2 的使能端，可以进行 PWM 调速，Pin7 是 M2 的正反转控制端。

注意：老版本的 RoMeo 电机控制端口为 Pin6、7、8、9，Pin6、9 为 PWM 调速端，Pin7、8 为电机正反转控制端。

演示代码：

```
/*
int E1 = 6;        //定义 M1 使能端
int E2 = 9;        //定义 M2 使能端
int M1 = 7;         //定义 M1 控制端
int M2 = 8;       //定义 M1 控制端
```

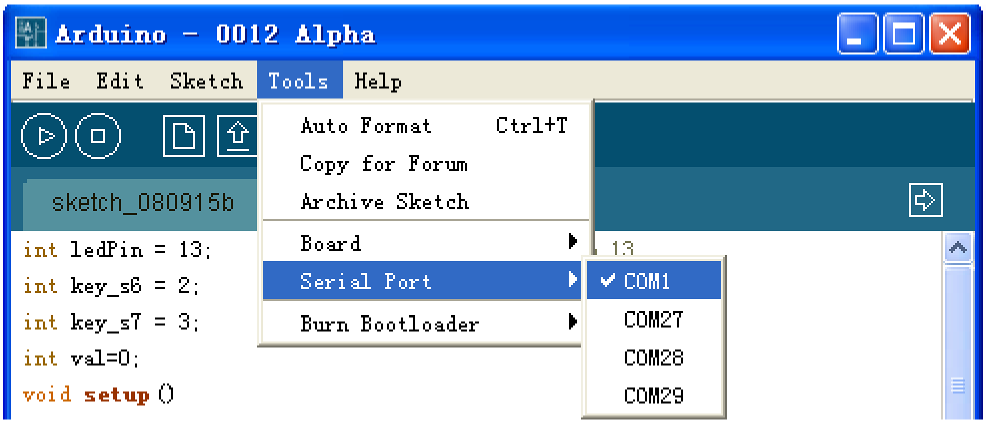
```
*/
int E1 = 5;         //定义 M1 使能端
int E2 = 6;         //定义 M2 使能端
int M1 = 4;        //定义 M1 控制端
int M2 = 7;        //定义 M1 控制端
void stop(void)                                    //停止
             {
                digitalWrite(E1,LOW);
                digitalWrite(E2,LOW);
             }
void advance(char a,char b)                  //前进
             {
                analogWrite (E1,a);            //PWM 调速
                digitalWrite(M1,HIGH);
                analogWrite (E2,b);
                digitalWrite(M2,HIGH);
             }
void back_off (char a,char b)                 //后退
             {
                analogWrite (E1,a);
                digitalWrite(M1,LOW);
                analogWrite (E2,b);
                digitalWrite(M2,LOW);
             }
void turn_L (char a,char b)                        //左转
             {
                analogWrite (E1,a);
                digitalWrite(M1,LOW);
                analogWrite (E2,b);
                digitalWrite(M2,HIGH);
             }
void turn_R (char a,char b)                        //右转
             {
                analogWrite (E1,a);
                digitalWrite(M1,HIGH);
                analogWrite (E2,b);
                digitalWrite(M2,LOW);
             }
void setup(void)
   {
      int i;
      for(i=6;i<=9;i++)
      pinMode(i, OUTPUT);
      Serial.begin(19200);          //设置串口波特率
   }
void loop(void)
```

```
{
      char val = Serial.read();
      if(val!=-1)
        {
           switch(val)
            {
              case 'w'://前进
                       advance (100,100);    //PWM 调速
                       break;
              case 's'://后退
                       back_off (100,100);
                       break;
              case 'a'://左转
                       turn_L (100,100);
                       break;
              case 'd'://右转
                       turn_R (100,100);
                       break;
             }
           delay(40);
        }
       else stop();
  }
```

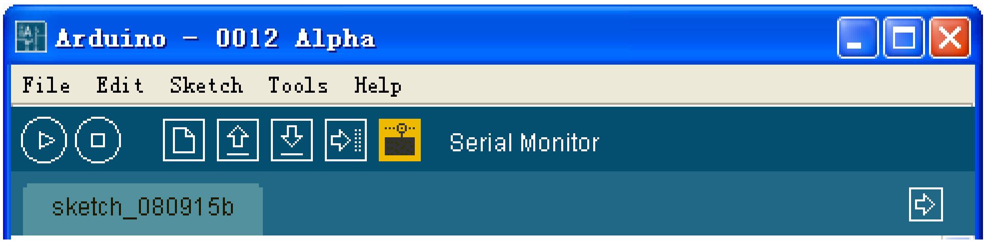
程序功能：使用 IDE 的串口助手发送字母，控制电机正转、反转或停止，并可以使用 PWM 调速，本程序可以直接用于控制小车前进、后退或转弯，速度可调整。

4. 串行端口的使用

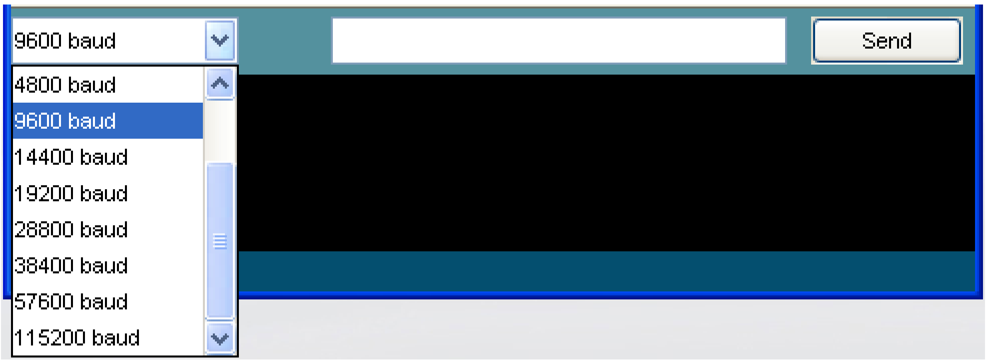
A. IDE 串口助手的使用



先要选择 RoMeo 的串口号，这步应该在开始实验时就完成了。



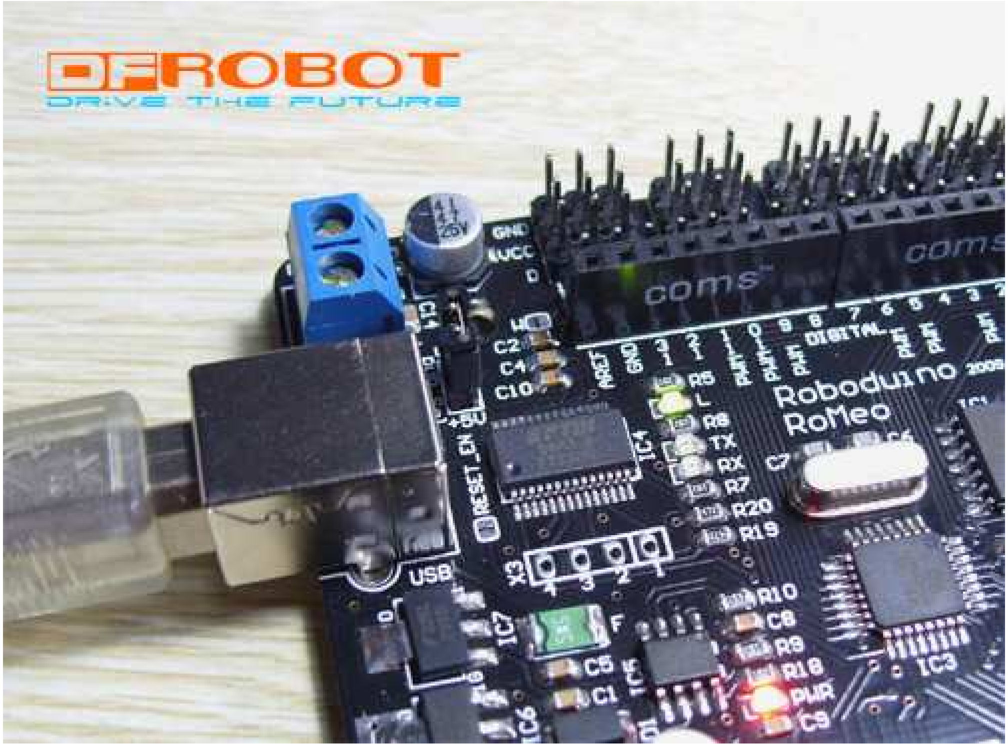
然后点上图中黄色的那个按钮（Serial Monitor）。



最后选择相应的波特率，右边的是发送文本框和发送按钮。

B．RoMeo 的串口输入

RoMeo 具有 1 个全双工 TTL 电平的串口，也就是数字端口的 Pin0（RX）和 Pin1（TX），程序的下载正是使用的这个串口，只不过被 USB 芯片将 TTL 串口转换成了 USB 接口而已。串口是个很常用的通讯端口，也是 RoMeo 最基本的通讯口，使用时需要注意端口电平和设置波特率。Arduino 语言提供了几个函数 Serial.begin ()、Serial.read()、Serial.print()、Serial.println()，实现串口的通讯操作，下面做个串口输入通讯的实验，通过 USB 接口和 RoMeo 串口通讯，将 USB 电缆插到 USB 口上并与电脑连接。

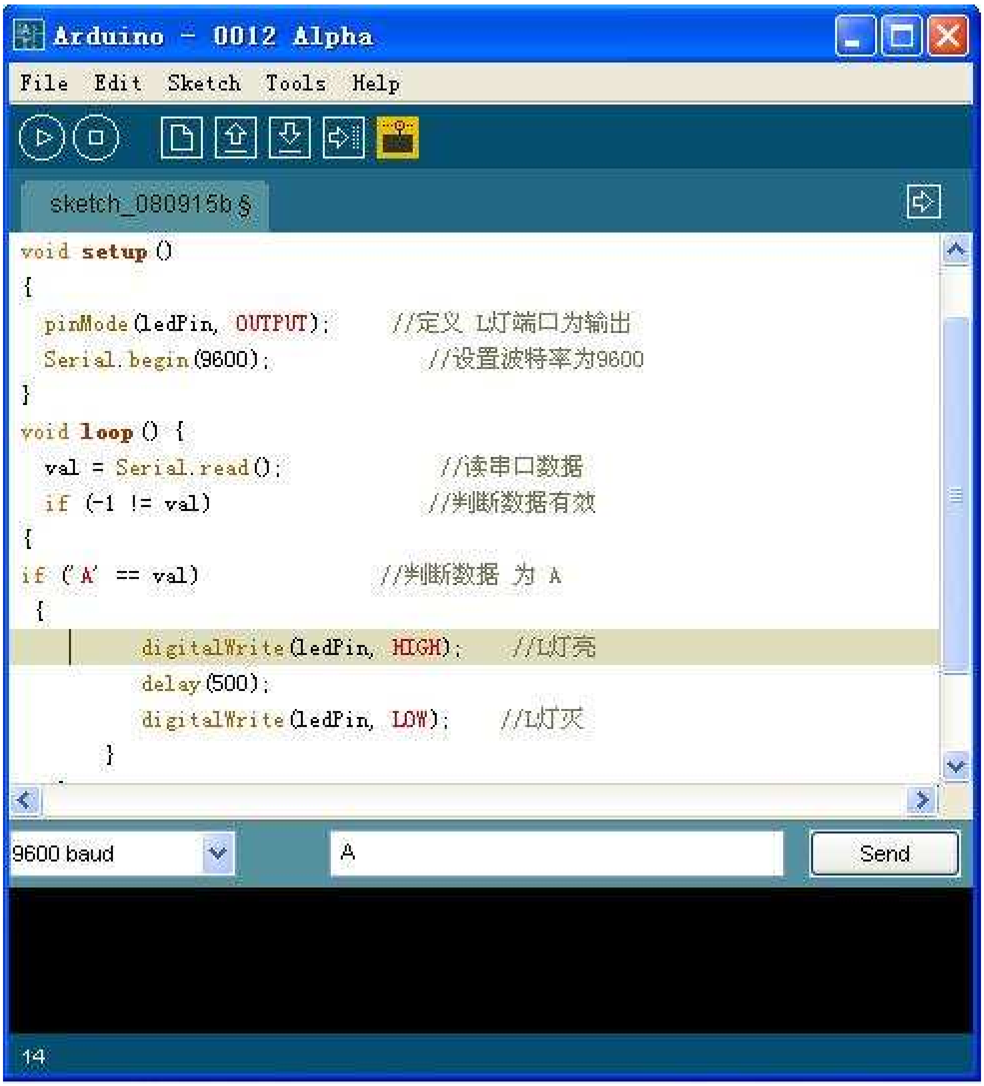


演示代码：

```
int ledPin = 13;
int val;
void setup()
{
  pinMode(ledPin, OUTPUT);      //定义 L 灯端口为输出
```

```
  Serial.begin(9600);              //初始化串口并设置波特率为 9600
}
void loop() {
  val = Serial.read();             //读串口数据
  if (-1 != val)                   //判断数据有效
   {
    if ('A' == val)                //判断数据  为  A
       {
           digitalWrite(ledPin, HIGH);      //L 灯亮
           delay(500);
           digitalWrite(ledPin, LOW);      //L 灯灭
       }
   }
}
```

程序功能：使用 IDE 的串口助手发送字母“A”，RoMeo 接收到后判断是不是“A”，如果是则让 L 灯闪烁一下，否则不进行任何操作。



C．RoMeo 的串口输出

演示代码：

```
void setup()
{
   Serial.begin(9600);              //初始化串口并设置波特率为 9600
}
void loop() {

   Serial.print("Hello!");          //串口向外发送字符串

   Serial.println(" DFRobot");      //串口向外发送字符串 ，并带回车

   delay(1000);                     //延时

}
```

程序功能： RoMeo 从串口间隙性的发送字符串“Hello！ DFRobot”。下面用 IDE 的串口助手观察结果。

D．RoMeo 串口无线数传（APC220）实验

RoMeo集成了一个TTL电平的无线数传接口，兼容APC220无线数传模块和DF-BluetoothV2蓝牙模块。

APC220无线数传模块的通信信道是半双工的，可以完成一点对一点，一点对多点的通讯。这二种方式首先需要设1个主站，其余为从站，所有站点都必须设置一个唯一的地址。通信的协调由主站控制，主站采用带地址码的数据帧发送数据或命令，所有从站全部都接收，并将接收到的地址码与本机地址码比较，地址不同则将数据丢掉，不做响应，若地址码相同，则将接收的数据传送出去。以上过程可以通过软件设置RFID Enable 自动实现，也可有用户通过上层协议完成。当设置RFID Enable时，APC220模块将自动比较所接收数据是否地址匹配，但不会自动应答，若地址匹配模块会将地址、数据传给终端设备。另外，组网必须保证在任何一个瞬间，同一个频点通信网中只有一个电台处于发送状态，以免相互干扰。APC220可以设置多个频道，所以可以在一个区域实现多个网络并存。

使用上面的串口输出代码演示，RoMeo 从串口间隙性的发送字符串“Hello！ DFRobot”，经过 APC220 发射到空中，被接到电脑上的 APC220 接收到。下面用 SSCOM3.2 串口助手观察结果。串口助手设置串口号为 APC220 的 COM4，波特率和程序设置的一致为 9600，其他默认即可，打开串口就可以看见结果。

注意：APC220 模块会占用串口端口，在下载程序的时候，请拔下 APC220 模块。

APC220 的连接方法

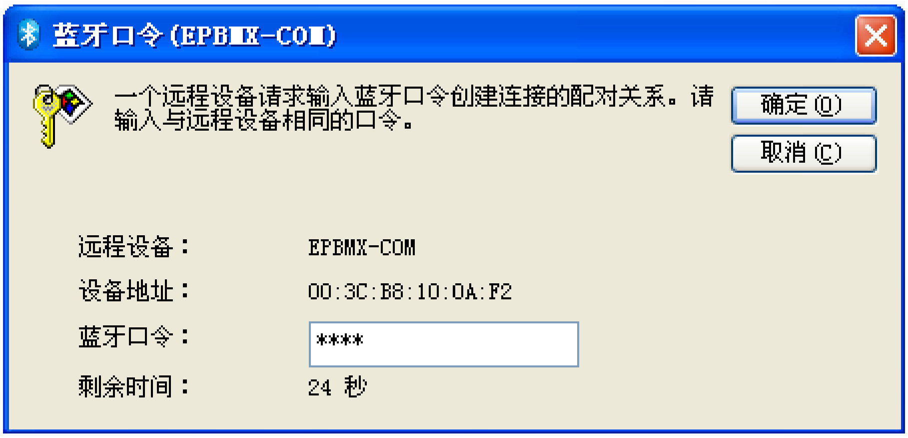
SSCOM3.2 串口助手观察的结果

E．RoMeo 串口蓝牙模块（DF-BluetoothV2）实验

RoMeo集成了一个TTL电平的无线数传接口，兼容APC220无线数传模块和DF-BluetoothV2蓝牙模块。

DF-BluetoothV2蓝牙模块基于蓝牙2.0规范，兼容蓝牙1.1、蓝牙1.2。DF-BluetoothV2蓝牙模块需要和蓝牙适配器配合使用。DF-BluetoothV2蓝牙模块只能做从机，蓝牙适配器做主机。使用蓝牙适配器需要安装一个管理软件IVT BlueSoleil（网上有下载），安装后，插上蓝牙适配器便会映射2个COM口出来，但不是我们需要的端口。

DF-BluetoothV2 蓝牙模块插到 RoMeo 上，蓝牙适配器插到电脑上，打开 IVT BlueSoleil 软件，黄色的球体就是蓝牙适配器，在黄色的球体上点右键→搜索设备，这是就会出现一个带问号的蓝色球体，这个就是我们的 DF-BluetoothV2 蓝牙模块了，在问号球体上点右键→配对，会出现下面的对话框，要求填写蓝牙口令，模块默认为四个零（0000），填好后点确定，配对后指示灯会一闪一闪的。

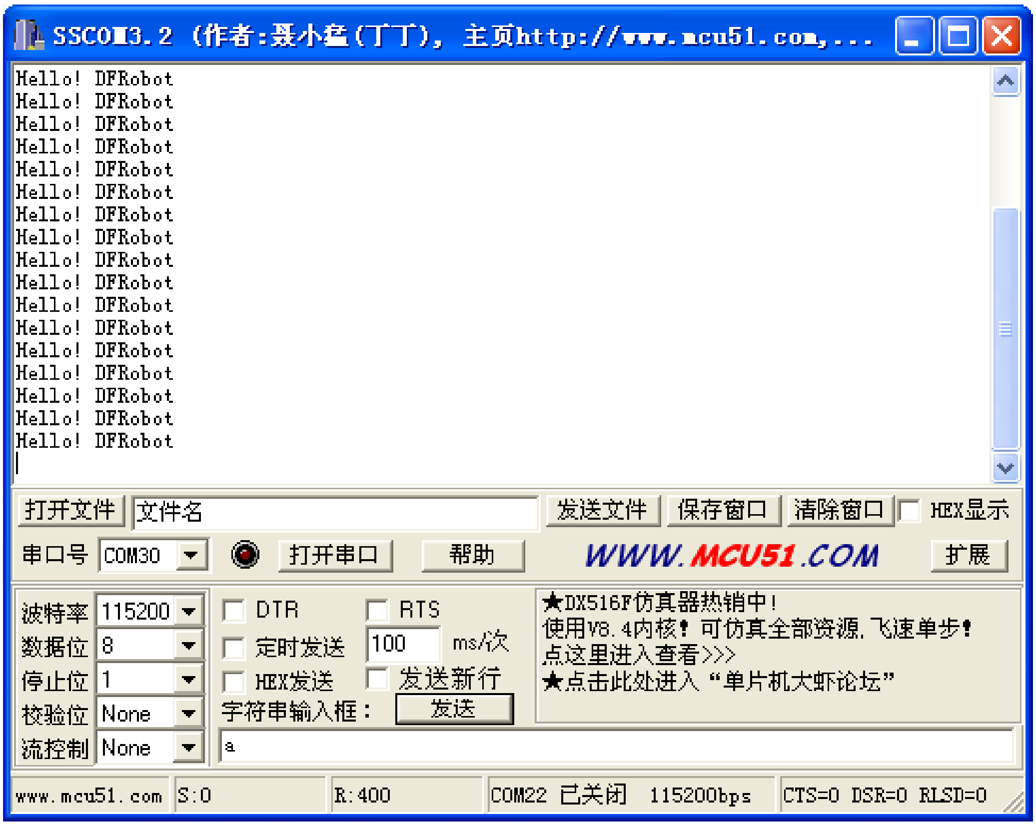


在问号球体上点右键→搜索服务，将会出现一个串口，接下来需要连接模块和适配器，在问号球体上点右键→连接蓝牙串口（如 COM30），这个新产生的 COM 端口号就是要用的串口号了。连接成功后，问号球体会由蓝色变为绿色，指示灯的闪烁也变为亮 1 秒灭 3 秒，并有虚线箭头线表示正在传输数据，还是使用之前的串口输出演示代码，但是这里波特率需要更改为 115200，必须和 DF-BluetoothV2 蓝牙模块的波特率（115200）一致。

注意：DF-BluetoothV2 蓝牙模块会占用串口端口，在下载程序的时候，请拔下 DF-BluetoothV2 蓝牙模块。

演示代码：

```
void setup()
{
  Serial.begin(115200);            //初始化串口并设置波特率为 115200
}
void loop() {
  Serial.print("Hello!");          //串口向外发送字符串
  Serial.println(" DFRobot");      //串口向外发送字符串 ，并带回车
  delay(1000);                     //延时
}
```



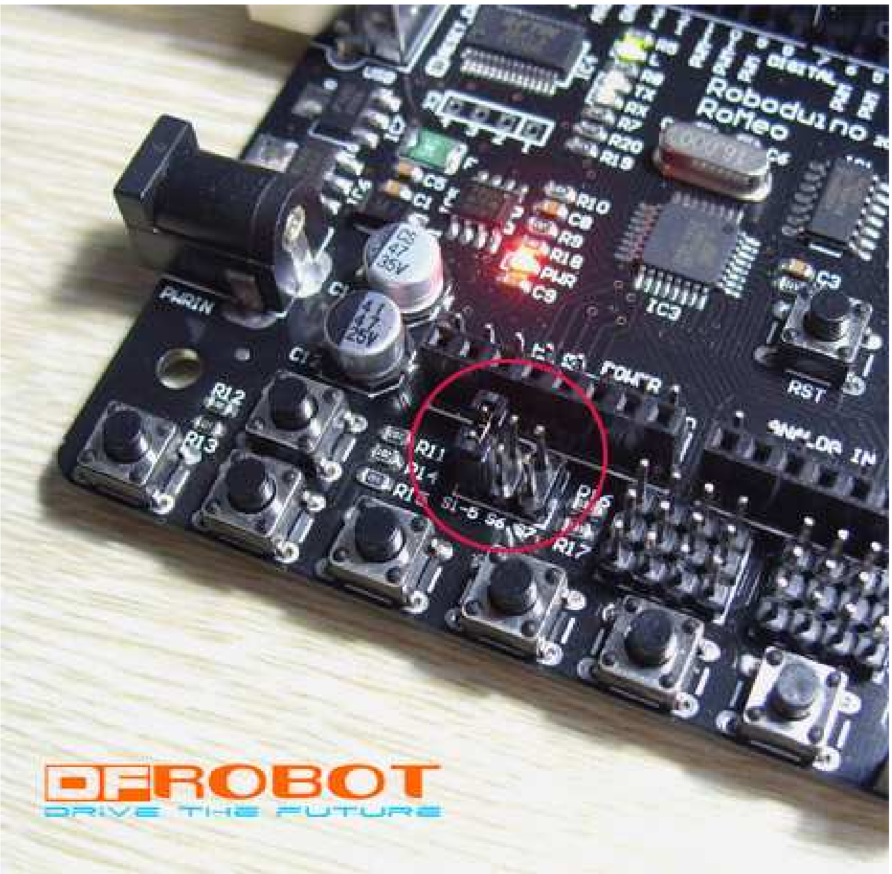
使用 SSCOM3.2 串口助手观察结果。串口助手设置串口号为 COM30，波特率和程序设置的一致为 115200，其他默认即可，打开串口就可以看见结果。

4. 模拟端口的使用

RoMeo 采用的 MEGA168-20AU，和 Arduino Nano 一样具有 8 个模拟口，排列方式与 Arduino 兼容。多出的 2 个模拟口放到了 POWER 的左侧。请使用 0012 以上版本 IDE，硬件选择 Arduino Nano。

A. 模拟按键实验

RoMeo 集成了 7 个按键 S1～S7，其中有 5 个按键（S1～S5）是使用模拟端口 7 控制。使用模拟按键需要设置跳线，如下图红圈中所示，将标有 S1-5 的跳线短接就选通，不用就断开。



演示代码 1：

```
int potPin = 7;            //定义输入模拟口 7
int ledPin = 13;           //定义 LED  为 Pin13
int val = 0;

void setup() {
   pinMode(ledPin, OUTPUT);    //设置 LED Pin13 为输出
}

void loop() {
   val = analogRead(potPin);       //读模拟口
   digitalWrite(ledPin, HIGH);    //点亮 LED
   delay(val);                    //延时
   digitalWrite(ledPin, LOW);     //关闭 LED
   delay(val);                    //延时
}
```

程序功能：分别按住 S1 到 S5 键，会看见 LED 闪烁的频率不同，这因为按键接入电阻不同，分到模拟口的电压就不同，AD 采集到的数据也就不同。

演示代码 2：

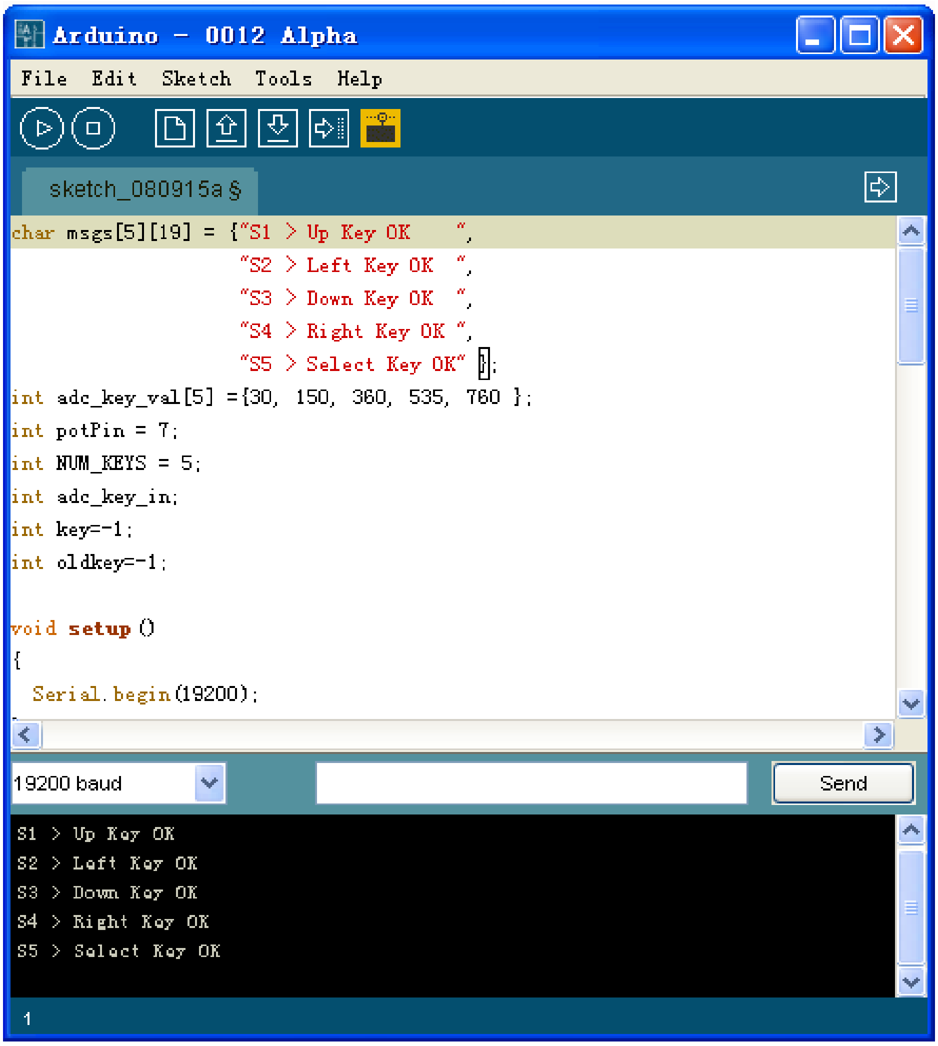
```
char msgs[5][19] = {      "S1 > Up Key OK        ",
                          "S2 > Left Key OK    ",
```

```
                              "S3 > Down Key OK    ",
                              "S4 > Right Key OK ",
                              "S5 > Select Key OK" };          //定义一个 2 维数组  存放显示字符
int adc_key_val[5] ={30, 150, 360, 535, 760 };                //定义一个数组  存放模拟键值比较值
int potPin = 7;                    //定义按键模拟端口 7
int NUM_KEYS = 5;
int adc_key_in;
int key=-1;
int oldkey=-1;

void setup()
{
   Serial.begin(19200);       //初始化串口  并设置波特率为 19200
}

void loop()
{
   adc_key_in = analogRead(potPin);                  //读模拟口 7
   key = get_key(adc_key_in);                        //调用判断按键程序
   if (key != oldkey)                                //  判断是否有新键按下
     {
        delay(50);                                   //  延时
        adc_key_in = analogRead(potPin);             //读模拟口 7
        key = get_key(adc_key_in);                   //调用判断按键程序
        if (key != oldkey)
          {
              oldkey = key;
              if (key >=0) Serial.println(msgs[key]);    //确认有键按下，就通过串口发送 2 维数组相应字符
          }
     }
  }
int get_key(unsigned int input)
{
       int k;
       for (k = 0; k < NUM_KEYS; k++)
         {
              if (input < adc_key_val[k]) return k;    //循环对比比较值，判断是否有键按下，有返回键号
         }
      if (k >= NUM_KEYS) k = -1;                      //没有键按下 k =-1
      return k;
}
```

程序功能：打开 IDE 串口助手，波特率选择 19200，分别按下 S1～S5，会看见 IDE 串口助手里显示按键对应的字符串。



B．GP2D12 输入实验

RoMeo 8 个模拟口对应有 8 个扩展插针，扩展口针脚排列根据 GP2D12 红外距离传感器设计，所以可以直接连接使用，GP2D12 是日本 SHARP 公司生产的红外距离传感器，测量距离 10～80cm 模拟输出，价格便宜，测距效果好，主要用于模型和机器人制作。它的输出模拟量与距离成反比，这点需要注意。

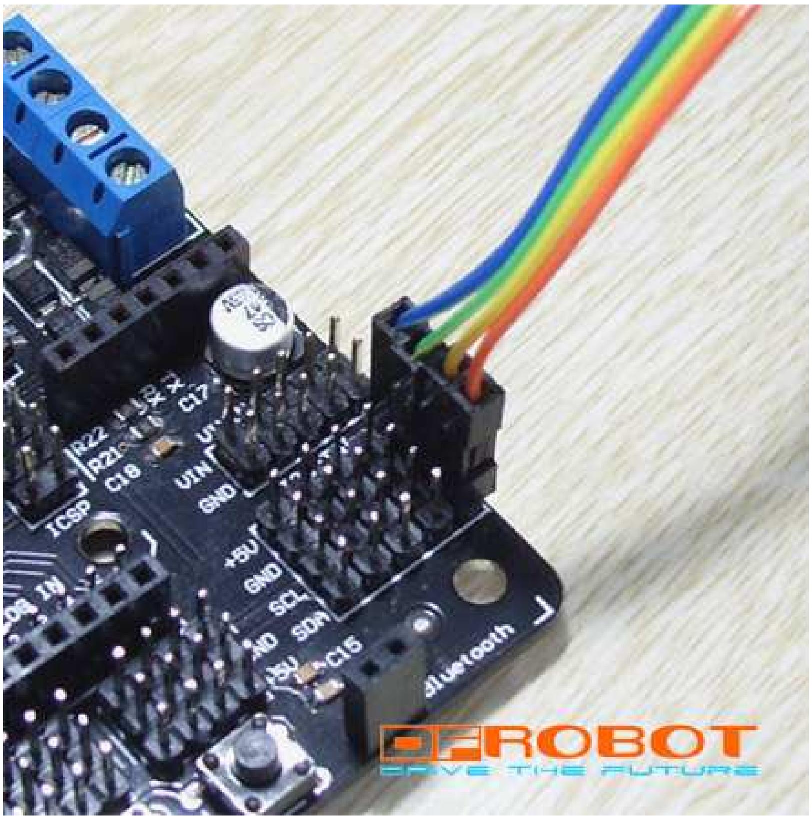
演示代码：

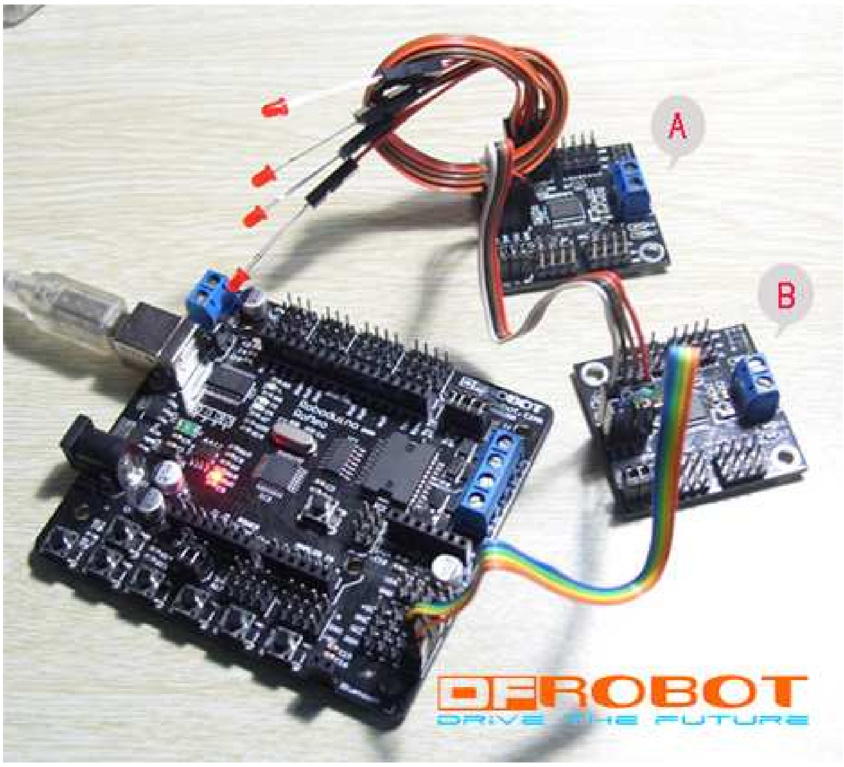
```
int GP2D12 = 0;              //定义输入模拟口 0
int val = 0;
void setup()
{
   Serial.begin(19200);      //初始化串口 并设置波特率为 19200
}
void loop()
{
   val = analogRead(GP2D12);       //读模拟口
   if(val!=-1) Serial.println(val);       //从串口发送采集到的模拟数据
   delay(100);
}
```

程序功能：打开 IDE 串口助手，波特率选择 19200，将手放到 GP2D12 正上方不同的高度，会看见 IDE 串口助手里显示的数据变化。

C．I2C 总线实验

RoMeo 的模拟口 4（SDA）和 5（SCL）具有 I2C 总线功能，我们设计时特意把 I2C 总线引出，做了 5 组插针，方便连接 I2C 器件。





RoMeo 与 I2C 转 16 路 GPIO 模块的连接，I2C 转 16 路 GPIO 模块可以扩展出 16 个双向 IO 口，I2C 总线上最多可以并接 8 个模块，每个模块采用跳线设置硬件地址，主机控制器程序也不复杂。模块 IO 口都具有内部上拉，可提供最大 50mA 漏电流和 20mA 灌电流。I2C 转 16 路 GPIO 模块的具体使用请参考：http://www.roboticfan.com/blog/user_2005/1229/archives/2008/200812802545.shtml

演示代码：

```
#include <Wire.h>       //库文件
//   地址          0 1 0 0   A2 A1 A0
#define KEY_ADDRESS 0x20 //模块 B 地址
#define LED_ADDRESS 0x27 //模块 A 地址
#define REGISTER_INPUT 0x00 //寄存器 0  设置 P0 输入
#define REGISTER_OUTPUT 0x02 //寄存器 2  设置 P0 输出参数
#define REGISTER_CONFIG 0x06 //寄存器 6  设置 P0 状态
void setup()
{
  Wire.begin();
  gpio_write(LED_ADDRESS, 0xffff);//设置模块 A  IO 口输出为高
  gpio_dir(LED_ADDRESS, 0x0000);//设置模块 A  IO 口为输出
}
void loop()
{
  int bits;
  bits = gpio_read(KEY_ADDRESS) & 0x0f;//读模块 B，因为输入寄存器
```

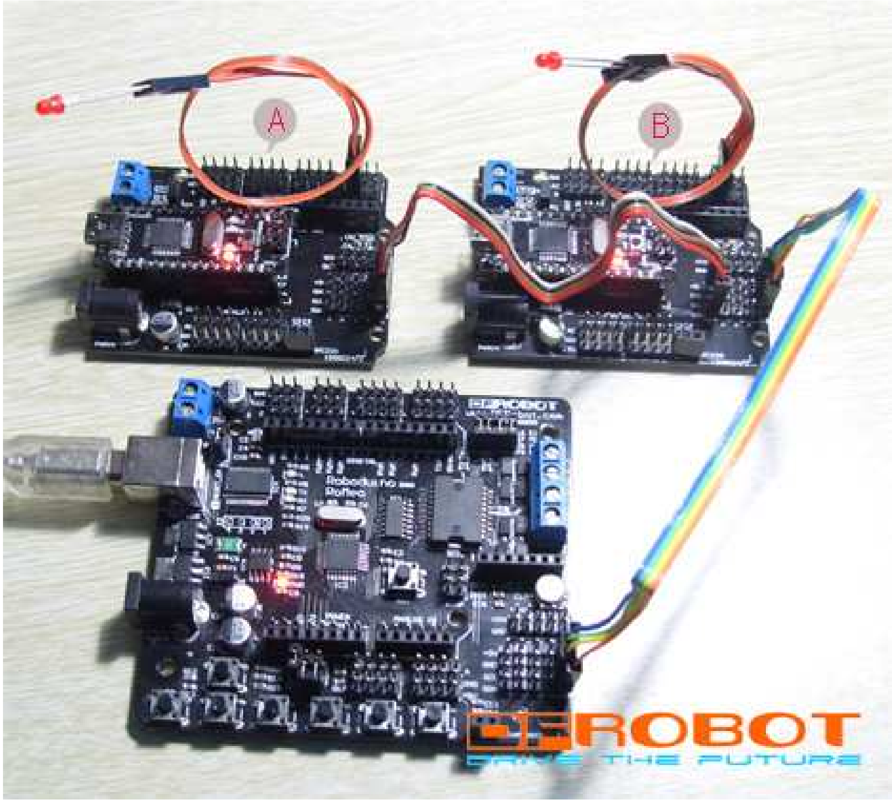
```
  gpio_write(LED_ADDRESS,
  (((bits & 1) << 3) | ((bits & 2) << 1) | ((bits & 4) >> 1) | ((bits & 8) >> 3)) << 12); // 将模块 B 的值镜像输出到模块 A
  delay(200);
}
int gpio_read(int address)//读函数
{
  int data = 0;

  // 发送输入寄存器地址
  Wire.beginTransmission(address);//模块地址
  Wire.send(REGISTER_INPUT);//寄存器地址
  Wire.endTransmission();

  // 连接模块并接收 2 字节数据
  Wire.beginTransmission(address);
  Wire.requestFrom(address, 2);
  if (Wire.available()) //读 P0
  {
data = Wire.receive();//读 P0 口 8 位
  }
  if (Wire.available())
 {
   data |= Wire.receive() << 8;//读 P1 口 8 位
  }
  Wire.endTransmission();
  return data;
}
void gpio_dir(int address, int dir) //配置函数
{
  // 发送配置寄存器地址
  Wire.beginTransmission(address);
  Wire.send(REGISTER_CONFIG);
  // 连接模块并发送 2 字节数据
  Wire.send(0xff & dir);  //  写 P0 配置寄存器 8 位
  Wire.send(dir >> 8);    //  写 P1 配置寄存器 8 位
  Wire.endTransmission();
}
void gpio_write(int address, int data) //写函数
{
  // 发送输出寄存器地址
  Wire.beginTransmission(address);
  Wire.send(REGISTER_OUTPUT);
  // 连接模块并发送 2 字节数据
  Wire.send(0xff & data);  //  写 P0 输出寄存器 8 位
```

```
  Wire.send(data >> 8);    //  写 P1 输出寄存器 8 位
  Wire.endTransmission();
}
```

程序功能：模块 A 做输出，P1 口高 4 位驱动 4 个 LED，模块 B 做输入，P0 口低 4 位接键盘。当模块 B 的 P0 口低 4 位有 1 位接地时，模块 A 的 P1 口高 4 位相反 IO 点亮一个 LED。



RoMeo 与 2 个 Nano 连接，组成多 CPU I2C 总线系统，RoMeo 做主机， 2 个 Nano 做从机，Nano 可单独编程，地址可任意设置。

RoMeo 主机演示代码：

```
#include <Wire.h>
void setup()
{
  Wire.begin();              //初始化 I2C 总线，不设地址表示默认为主机
  Serial.begin(9600);      //
  Serial.println("Ready");
}
void loop()
{
  int val;
  if(Serial.available() > 0)
   {
     val=Serial.read();
```

```
    if(val==49)        //1
     {
       Wire.beginTransmission(4); //  从机地址#4
       Wire.send(1);                      //  发送字符 1
       Wire.endTransmission();     //  停止传送
       Serial.println("49 OK");
       delay(10);
     }
    else if(val==50)         //2
     {
       Wire.beginTransmission(4); //  从机地址#4
       Wire.send(0);                      //  发送字符 0
       Wire.endTransmission();     //停止传送
       Serial.println("50 OK");
       delay(10);
     }
    else if(val==51)       //3
     {
       Wire.beginTransmission(5); //  从机地址#5
       Wire.send(1);                      //  发送字符 1
       Wire.endTransmission();     //停止传送
       Serial.println("51 OK");
       delay(10); //:P
     }
    else if(val==52)       //4
     {
       Wire.beginTransmission(5); //  从机地址#5
       Wire.send(0);                      //  发送字符 0
       Wire.endTransmission();     //停止传送
       Serial.println("52 OK");
       delay(10);
     }
     else Serial.println(val);
  }
}
```

Nano 从机 A 演示代码：

```
#include <Wire.h>
int LED = 2;
void setup()
{
  Wire.begin(4);                          //  设置从机地址为#4
  Wire.onReceive(receiveEvent); //
```

```
  pinMode(LED,OUTPUT);
}
void loop()
{
  delay(100);
}

void receiveEvent(int howMany)
{
    int c = Wire.receive();         //  接收字符
    if(c==1)
      {
        digitalWrite(LED,HIGH);
      }
    else if(c==0)
      {
        digitalWrite(LED,LOW);
      }
}
```

Nano 从机 B 演示代码：

```
#include <Wire.h>
int LED = 2;
void setup()
{
  Wire.begin(5);  //  设置从机地址为#5
  Wire.onReceive(receiveEvent); //
  pinMode(LED,OUTPUT);
}
void loop()
{
  delay(100);
}

void receiveEvent(int howMany)
{
    int c = Wire.receive();    //  接收字符
    if(c==1)
      {
        digitalWrite(LED,HIGH);
      }
    else if(c==0)
      {
        digitalWrite(LED,LOW);
```

```
    }
}
```

程序功能：将 3 个代码分别编译后，分别下载到 3 个控制器中，然后打开 IDE 的串口助手，从串口助手的文本框中分别发送数字 1，2，3，4 来到 RoMeo 主机，RoMeo 主机再通过 I2C 总线将指令分别转发给从机 A 和 B，从机 A 和 B 接收到主机发送来的 1 点亮 LED，接收到 0 关掉 LED。

本手册中的演示代码均通过验证！

本手册版权归 DFRobot 所有！

发布日期	版本号	备注
2009 年 4 月 10 日	**V1.0**	**建文档**
2009 年 8 月 26 日	**V1.1**	**更改例程错误**
2009 年 9 月 24 日	**V1.2**	**更新插图**
2010 年 3 月 9 日	**V1.3**	**Romeo 硬件修改电机控制端口**

Copyright by DFRobot